

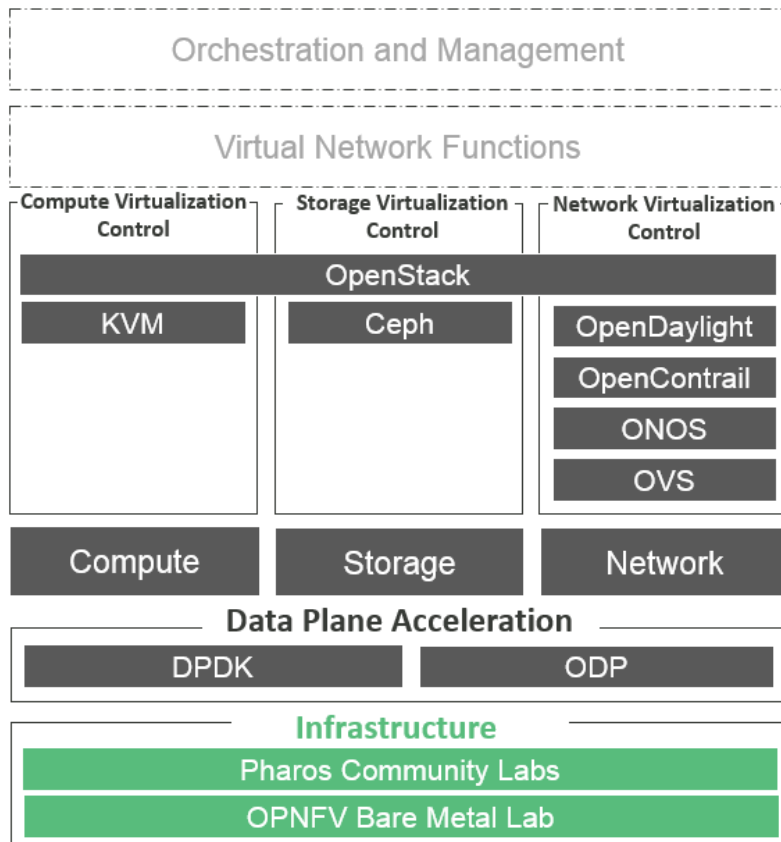
OPNFV

(Open Platform for NFV)

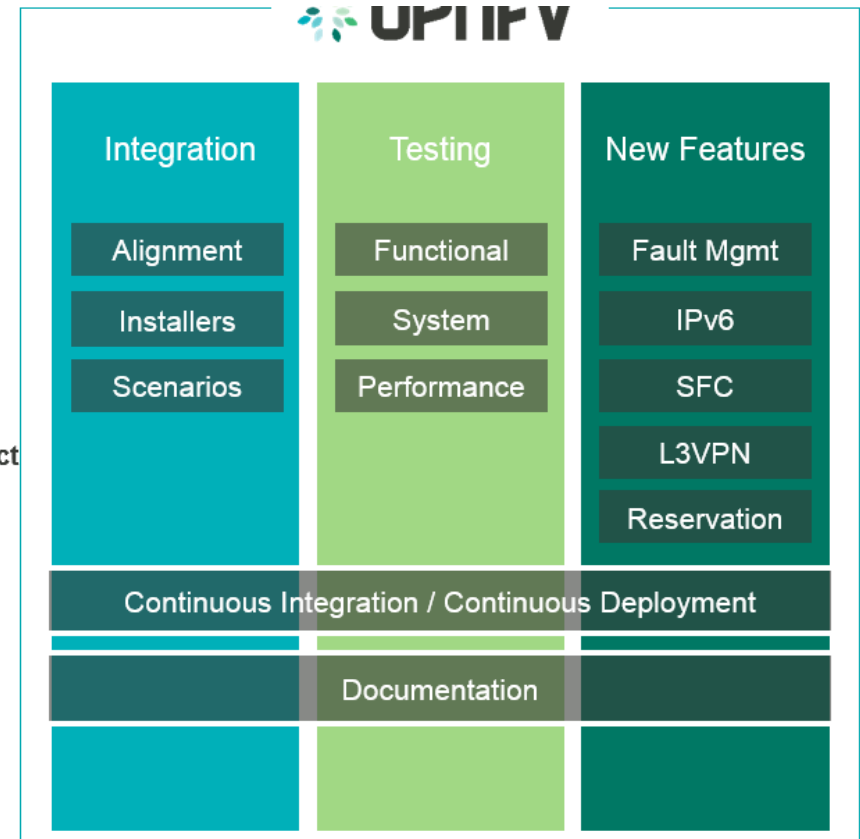
송실대학교
김영한

OPNFV review: History and Achievement

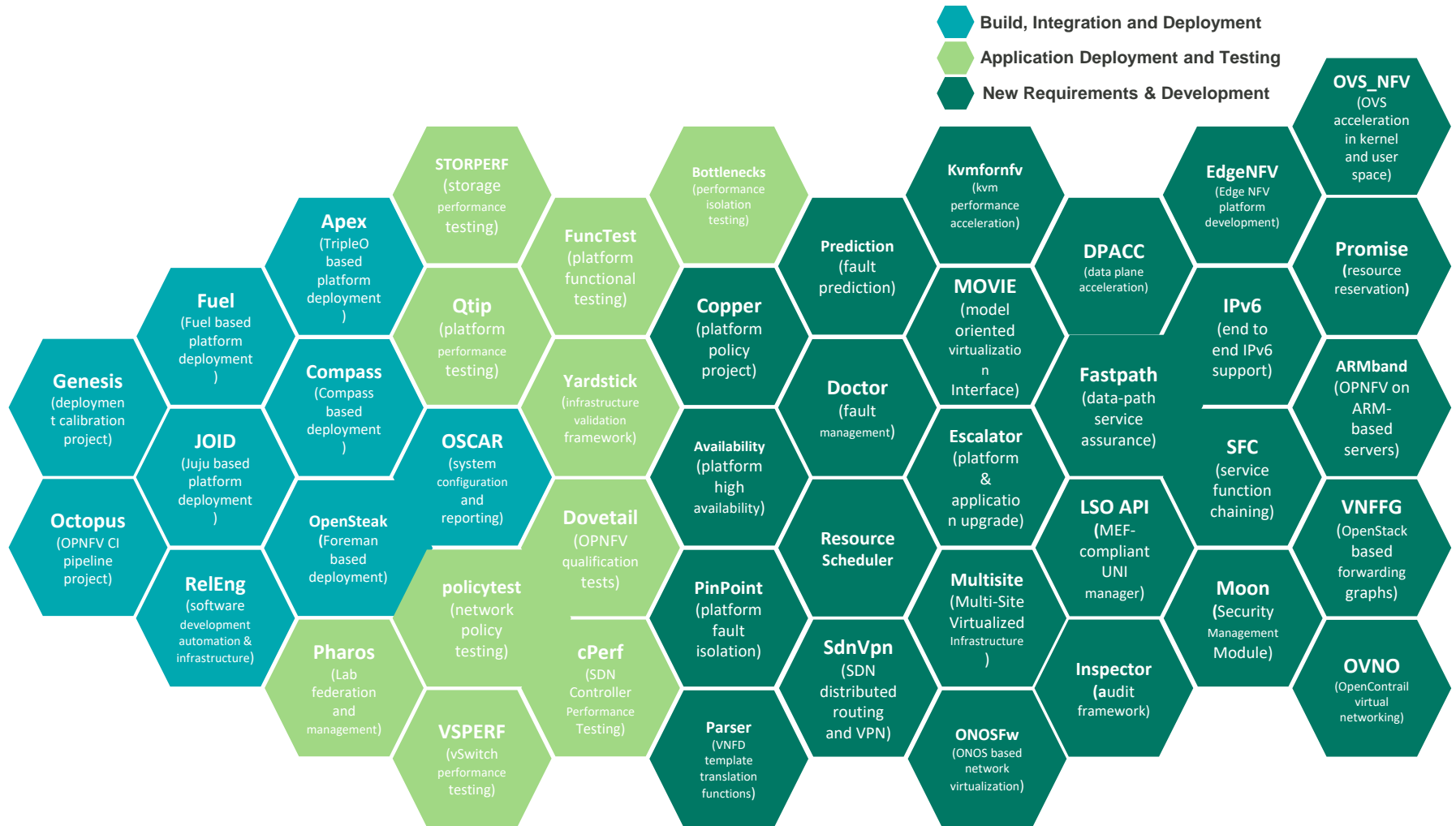
OPNFV Platform Overview



Upstream Project Collaboration:



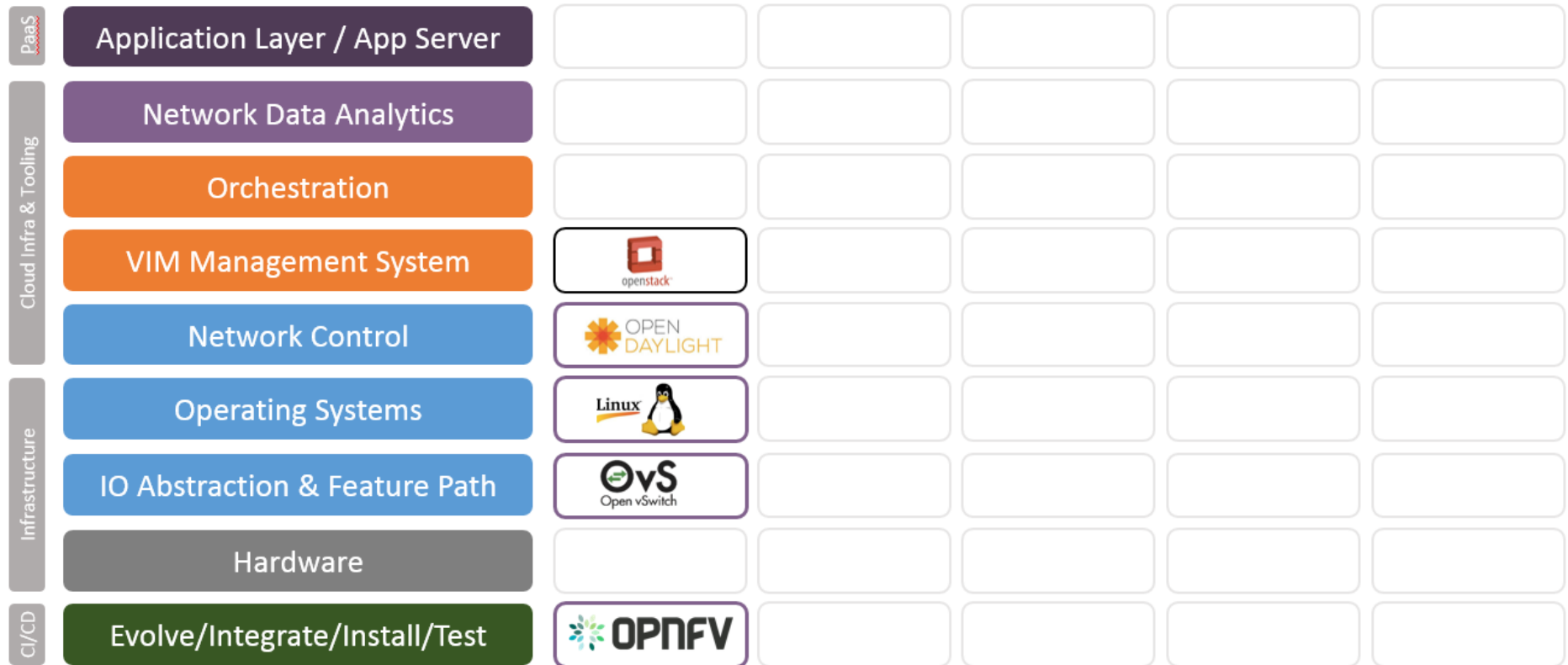
OPNFV Project pipeline



OPNFV project goals

- Develop an integrated and tested open source platform that can be used to build NFV functionality-accelerating the introduction of new products and services.
- Include participation of leading end users to validate that OPNFV meets the needs of user community
- Contribute to and participate in relevant open source projects that will be leveraged in the OPNFV platform; ensuring consistency, performance and interoperability among open source components
- Establish an ecosystem for NFV solutions based on open standards and software to meet the needs of end users
- Promote OPNFV as the preferred platform and community for open source NFV

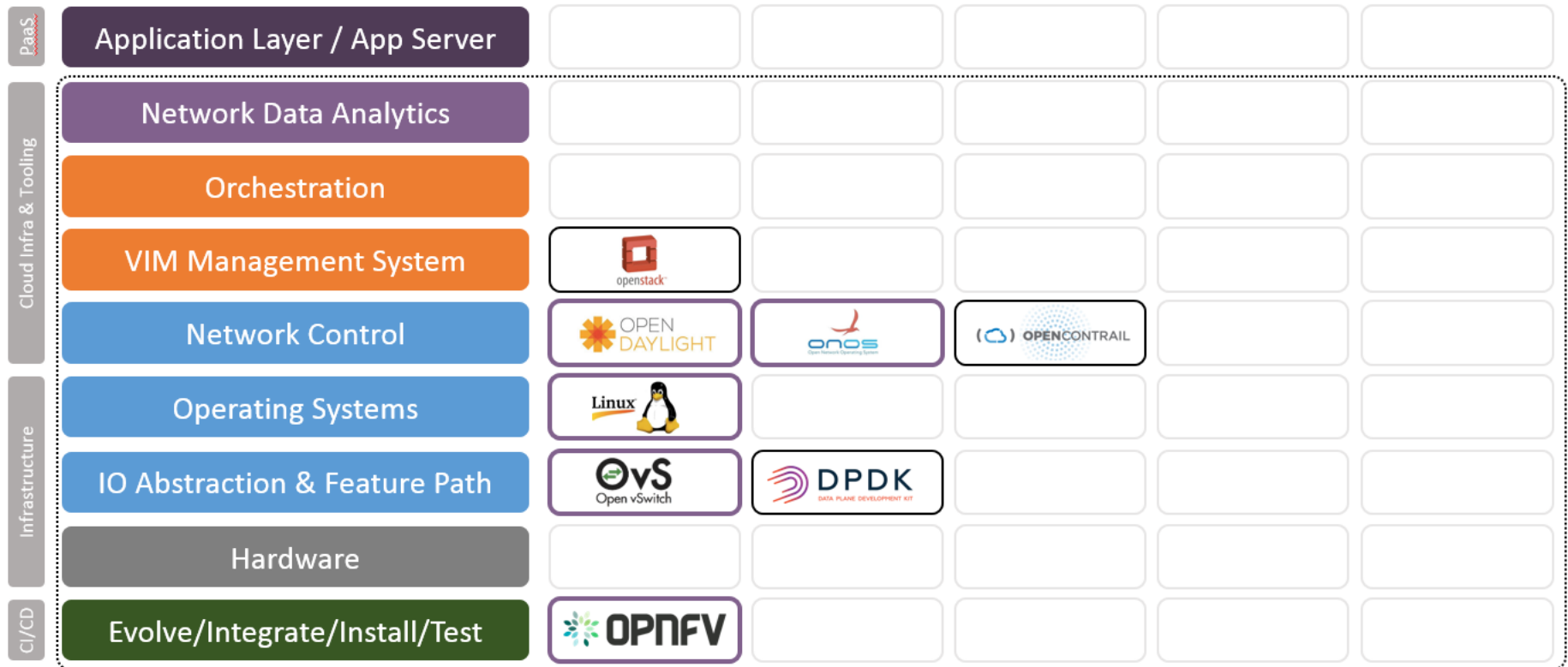
OPNFV overview: OPNFV Arno Release



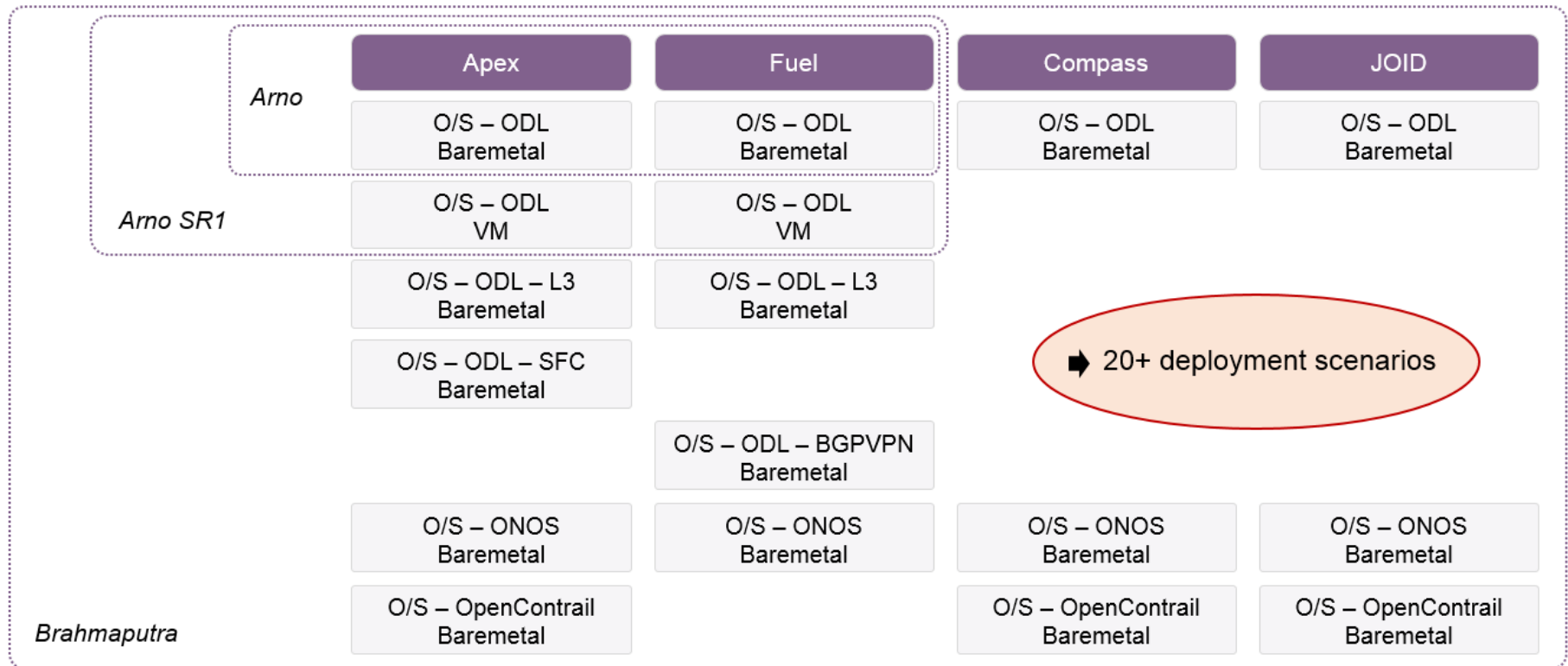
OPNFV overview

- **5 Projects Participated in Arno:**
 - Lab Infrastructure:
“Pharos”
 - Create and maintain a system-level CI/CD system:
“Octopus”
 - System Composition and Installation:
“Bootstrap/GetStarted”
 - Functional Testing:
“FuncTest”
 - Documentation:
“OPNFV Docs”

OPNFV overview: OPNFV Brahmaputra

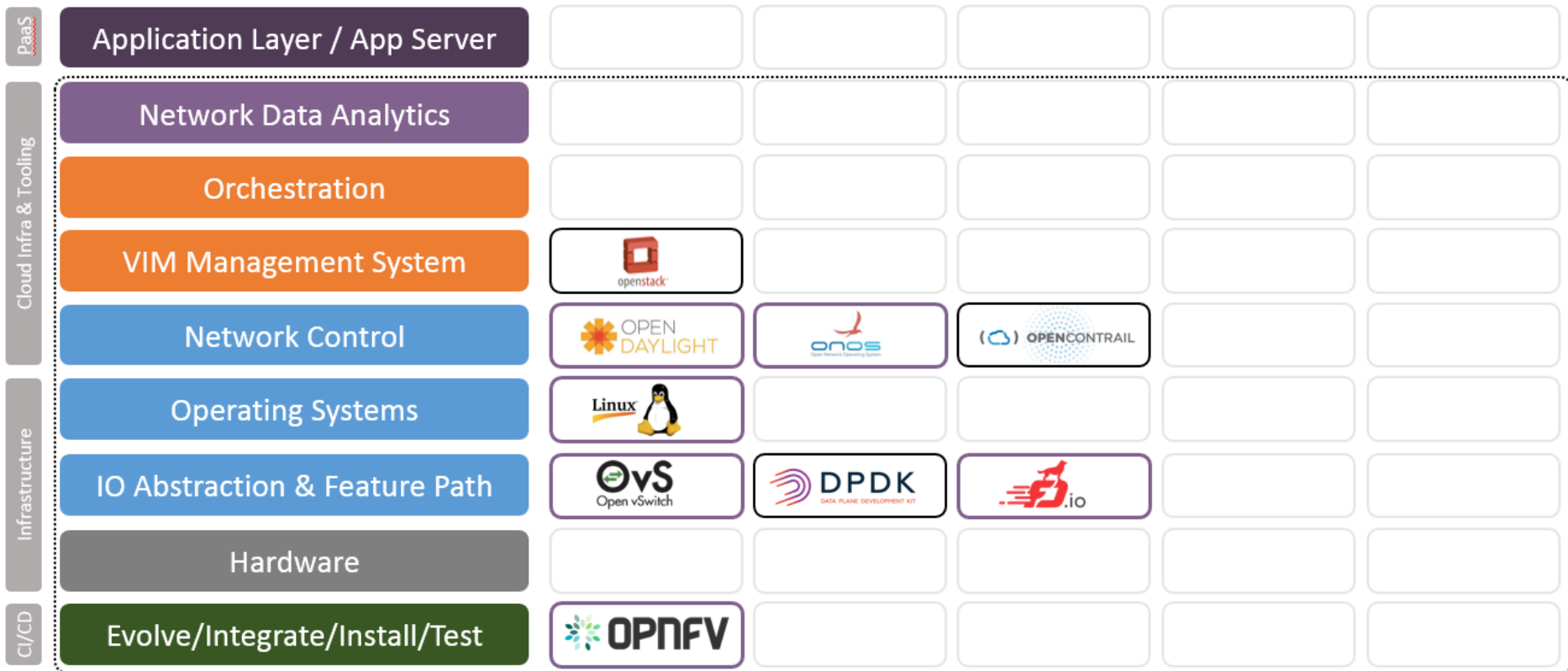


OPNFV overview: OPNFV Brahmaputra



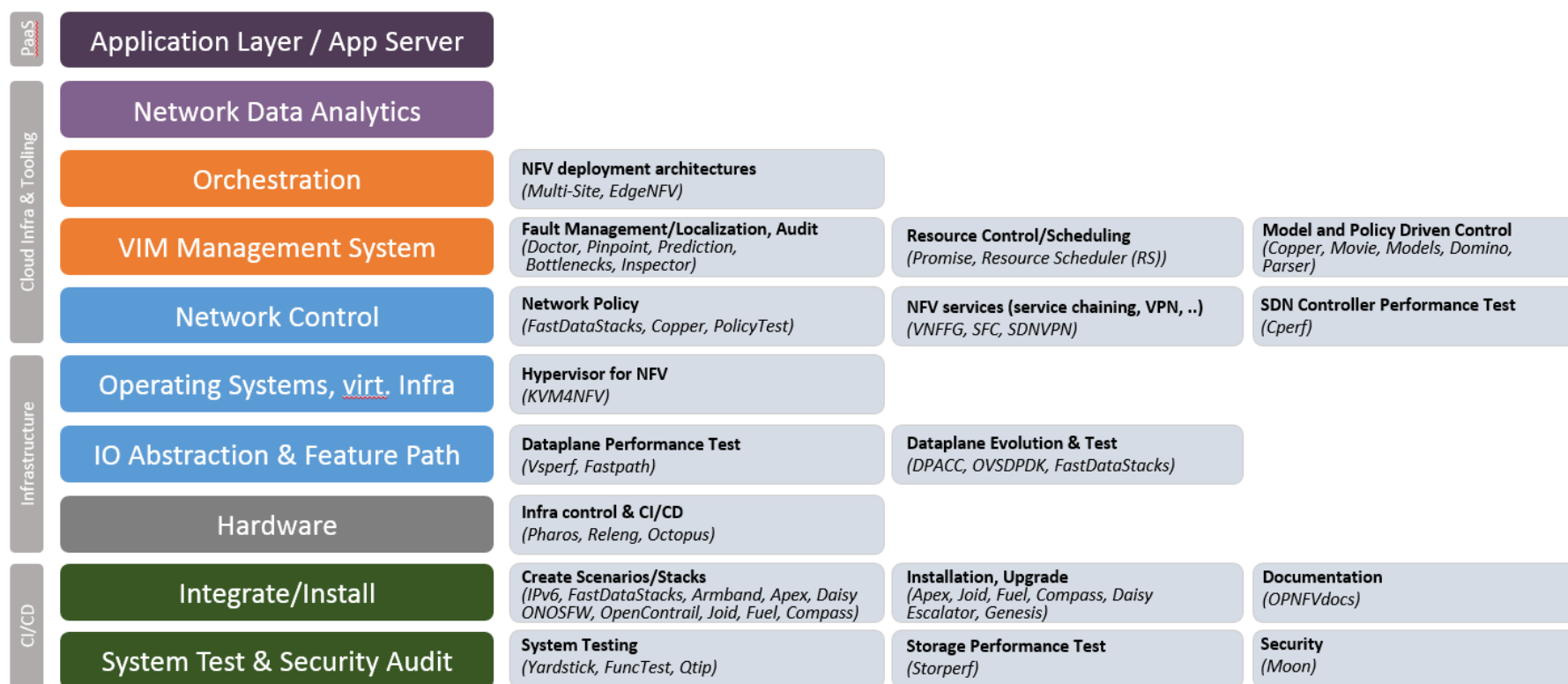
Note: Not all scenarios shown. For an up to date view on scenarios see <https://build.opnfv.org/ci/view/OPNFV%20Platform%20CI%20-%20Alternative%20View/>

OPNFV overview: OPNFV Colorado

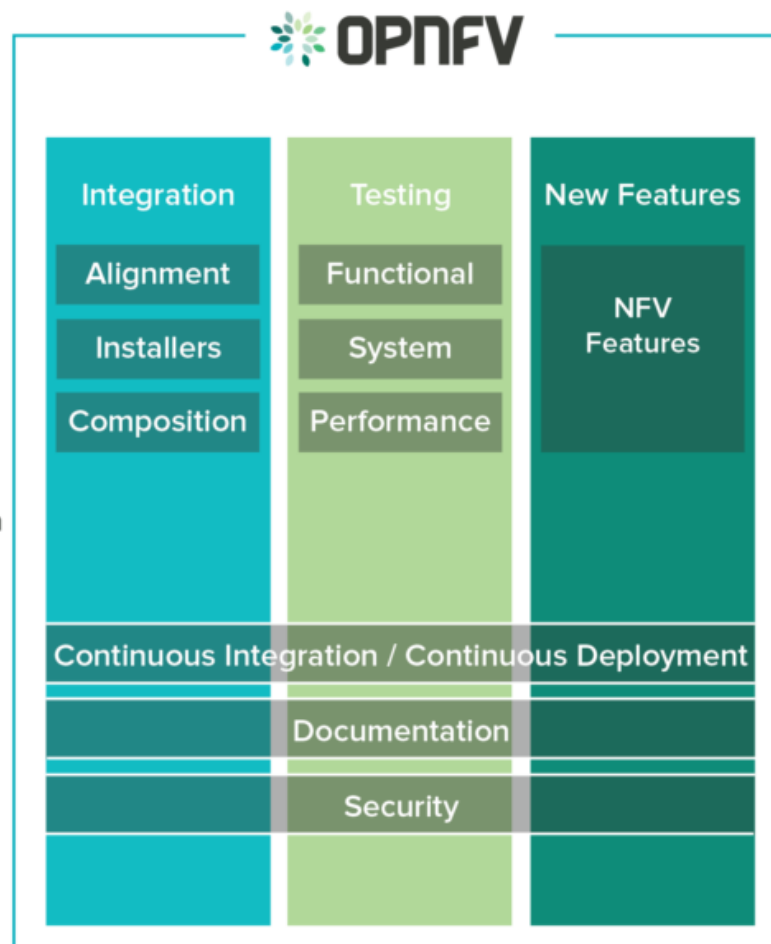
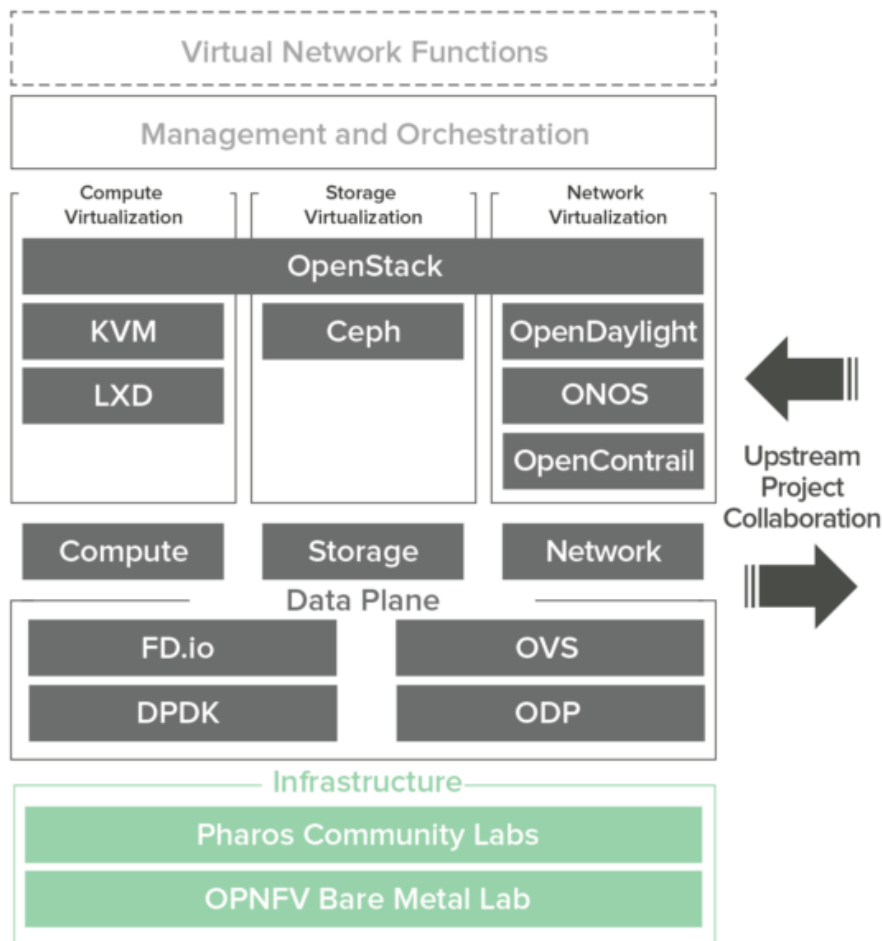


OPNFV overview: OPNFV Colorado

- A map of OPNFV projects



OPNFV Danube 1.0



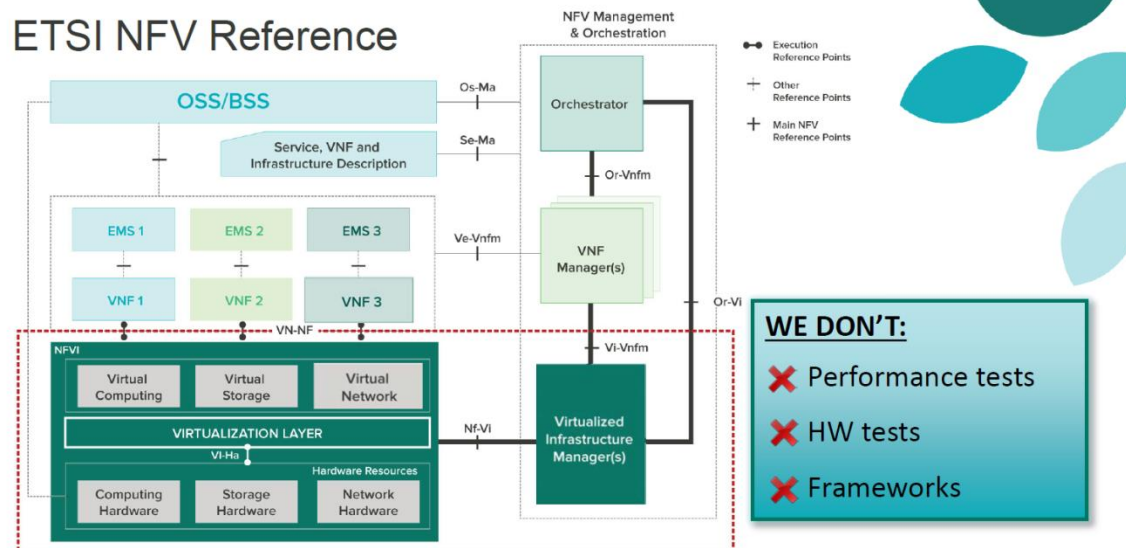
Key Features

- MANO
 - Integration between NFV Infrastructure/Virtual Infrastructure Manager (NFVI/VIM) with Open-Orchestration (Open-O) platform (now ONAP)
 - Support Service Assurance and other use cases; multi-domain template support (Domino project)
 - Translation features between YANG and Tosca modeling languages
- Enhanced DevOps automation and testing methodologies
 - bring a fully integrated CI/CD pipeline
 - the introduction of stress testing into the OPNFV test suite
 - <http://testresults.opnfv.org/reporting/danube.html>
- Focus on NFV performance
 - acceleration of the data plane via FD.io integration for all Layer 2 and Layer 3 forwarding
- Feature enrichment and hardening
 - core NFVI/VIM functionality
 - such as IPv6, Service Function Chaining (SFC), L2 and L3 Virtual Private Network (VPN), fault management and analysis,

OPNFV testing projects: OPNFV FuncTest

OPNFV FuncTest

- Focus on verifying the OpenStack deployment and the SDN Controllers
- Having a full integration and automation mechanism
- Provide comprehensive testing methodology



Test Cases

vPing test case

Create machines and verify connectivity



ODL test case

Robot framework, ODL functional testing



Rally bench tests

Benchmark the OpenStack deployment



Tempest test

OpenStack native tests (100+ smoke-tests)



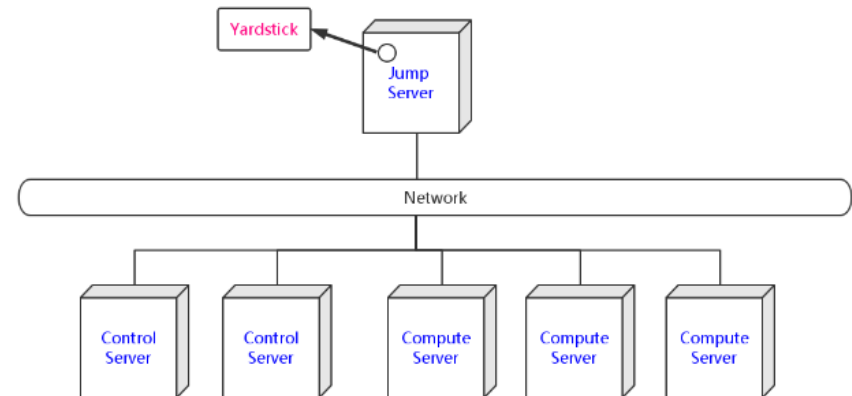
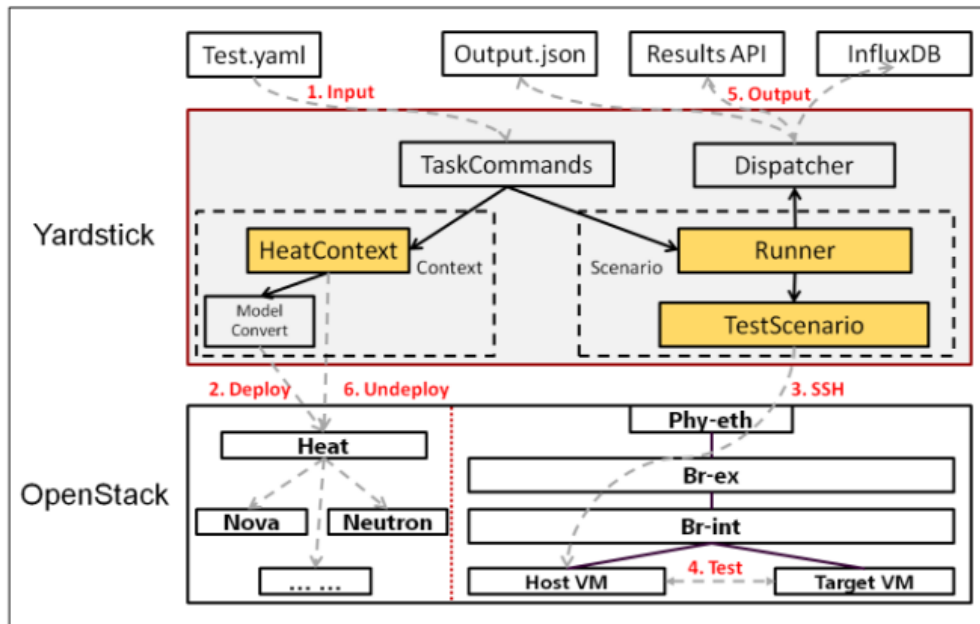
OPNFV testing projects: OPNFV YardStick

YardStick Methodology

- Step1: Define Infrastructure
 - the Hardware, Software and corresponding configuration target for validation; the OPNFV infrastructure, in OPNFV community labs.
- Step2: Identify VNF type
 - the application for which the infrastructure is to be validated, and its requirements on the underlying infrastructure.
- Step3: Select test cases
 - depending on the workload that represents the application for which the infrastructure is to be validated, the relevant test cases amongst the list of available Yardstick test cases.
- Step4: Execute tests
 - define the duration and number of iterations for the selected test cases, tests runs are automated via OPNFV Jenkins Jobs.
- Step5: Collect results
 - using the common API for result collection.

Architecture

- Logical view and deployment view



YardStick use cases:

Performance/Speed Metrics

Category	Performance/Speed
Compute	<ul style="list-style-type: none">•Latency for random memory access•Latency for cache read/write operations•Processing speed (instructions per second)•Throughput for random memory access (bytes per second)
Network	<ul style="list-style-type: none">•Throughput per NFVI node (frames/byte per second)•Throughput provided to a VM (frames/byte per second)•Latency per traffic flow•Latency between VMs•Latency between NFVI nodes•Packet delay variation (jitter) between VMs•Packet delay variation (jitter) between NFVI nodes
Storage	<ul style="list-style-type: none">•Sequential read/write IOPS•Random read/write IOPS•Latency for storage read/write operations•Throughput for storage read/write operations

YardStick use cases: Capacity/Scale Metrics

Category	Capacity/Scale
Compute	<ul style="list-style-type: none">•Number of cores and threads- Available memory size•Cache size•Processor utilization (max, average, standard deviation)•Memory utilization (max, average, standard deviation)•Cache utilization (max, average, standard deviation)
Network	<ul style="list-style-type: none">•Number of connections•Number of frames sent/received•Maximum throughput between VMs (frames/byte per second)•Maximum throughput between NFVI nodes (frames/byte per second)•Network utilization (max, average, standard deviation)•Number of traffic flows
Storage	<ul style="list-style-type: none">•Storage/Disk size•Capacity allocation (block-based, object-based)•Block size•Maximum sequential read/write IOPS•Maximum random read/write IOPS•Disk utilization (max, average, standard deviation)

YardStick use cases:

Availability/Reliability Metrics

Category	Availability/Reliability
Compute	<ul style="list-style-type: none">•Processor availability (Error free processing time)•Memory availability (Error free memory time)•Processor mean-time-to-failure•Memory mean-time-to-failure•Number of processing faults per second
Network	<ul style="list-style-type: none">•NIC availability (Error free connection time)•Link availability (Error free transmission time)•NIC mean-time-to-failure•Network timeout duration due to link failure•Frame loss rate
Storage	<ul style="list-style-type: none">•Disk availability (Error free disk access time)•Disk mean-time-to-failure•Number of failed storage read/write operations per second

MANO-related Projects in OPNFV

MANO-related projects in OPNFV

- Open-O (**Opera**)
- Open Source Mano(**OSM**)
- OpenBaton (**Orchestra**)
- JuJu (**JOID**)
- OpenStack Tacker

OPNFV vs CORD

- **Central Office Re-architected as a Datacenter (CORD)** is also an open source NFV project.
- CORD's scope is focused, it only deals with the telco Central Offices and the task of converting them to agile clouds that can host NFV workloads for mobile, enterprise and residential use cases.
- CORD develops a POD consisting of common hardware and software (see diagram below) and customizes it for these three use cases by changing access and core connectivity, and VNF services.
- Unlike OPNFV that promotes technology choices, CORD limits technology choices for different layers of technology and is therefore fairly prescriptive.



OPNFV Installation Tutorial

Contents

- OPNFV Danube Review
 - Danube release
 - key enhancements
 - Scenarios in danube
- OPNFV Installation
 - OPNFV Apex review
- OPNFV Apex Installer Demo

OPNFV Danube Review

OPNFV Danube Review

- OPNFV Danube brings
 - **DevOps methodologies** to NFV via collaborative upstream development,
 - integration, deployment
 - significant Continuous Integration/Continuous Development **(CI/CD) testing automation.**
- Danube also includes continued
 - incremental improvements in **baseline features**,
 - new advances in **network control capabilities**,
 - growing focus on performance-related topics—especially NFV **data plane performance**
 - instrumentation around **MANO functions.**

OPNFV Danube Review-key enhancements

- **Foundational support and introduction of capabilities for MANO:**
 - Integration between NFV Infrastructure/Virtual Infrastructure Manager (NFVI/VIM) **with Open-Orchestration (Open-O) platform (now ONAP)**;
 - instrumentation of NFVI **network telemetry to support Service Assurance** and other use cases;
 - **multi-domain template** support (Domino project);
 - **translation features** between YANG and Tosca modeling languages (Parser project).

OPNFV Danube Review-key enhancements

- **Enhanced DevOps automation and testing methodologies**
 - bring a fully integrated **CI/CD pipeline**
 - the creation of **Lab-as-a-Service (LaaS)** to enable dynamic provisioning of lab resources
 - the introduction of **stress testing** into the OPNFV test suite
 - **Common Dashboard** that provides a consistent view of the testing ecosystem

OPNFV Danube Review-key enhancements

- **Focus on NFV performance**

- acceleration of the data plane via **FD.io integration** for all Layer 2 and Layer 3 forwarding (FastDataStacks project)
- continued enhancements to **OVS-DPDK and KVM**
- The release also sees a renewed focus on **performance test project** activities **through virtual switch testing** (VSPERF project)
- root cause analysis for platform **performance issues** (Bottlenecks)
- initial compute subsystem performance testing to lay the groundwork for **Benchmarking As a Service** (QTIP project)
- **storage subsystem performance** testing (Storperf project).

OPNFV Danube Review-key enhancements

- **Key NFV architectural enhancements**
 - including the ability to dynamically enable and configure network control through integration with **OpenStack Gluon**
 - increased reliability and test cases that support **multi-site and High Availability (HA) work**

OPNFV Danube Review-key enhancements

- **Feature enrichment and hardening**
 - core NFVI/VIM functionality such as IPv6
 - Service Function Chaining (SFC)
 - L2 and L3 Virtual Private Network (VPN)
 - fault management and analysis

OPNFV Danube Review

- Danube Scenario Status

Scenario	Installer	Owner	Jenkins Job Created (Y/N)	Bugs	Intent to Release (Y/N) 1.0	Intent to Release (Y/N) 2.0	Intent to Release (Y/N) 3.0
os-nosdn-nofeature-ha	Apex	@Tim Rozet	Y		Y	Y	Y
os-odl_I3-nofeature-ha	Apex	@Tim Rozet	Y		Y	Y	Y
os-onos-nofeature-ha	Apex	@Tim Rozet	Y		N	Y	Y
os-odl-bgpvpn-ha	Apex	@Tim Irmich	Y		N	Y	Y
os-odl_I2-sfc-fdio-ha	Apex	@Brady Johnson	N		N	N	N
os-odl_I2-sfc-ha	Apex	@Brady Johnson			N	Y	Y
os-odl_I2-sfc-noha	Apex	@Brady Johnson			N	Y	Y

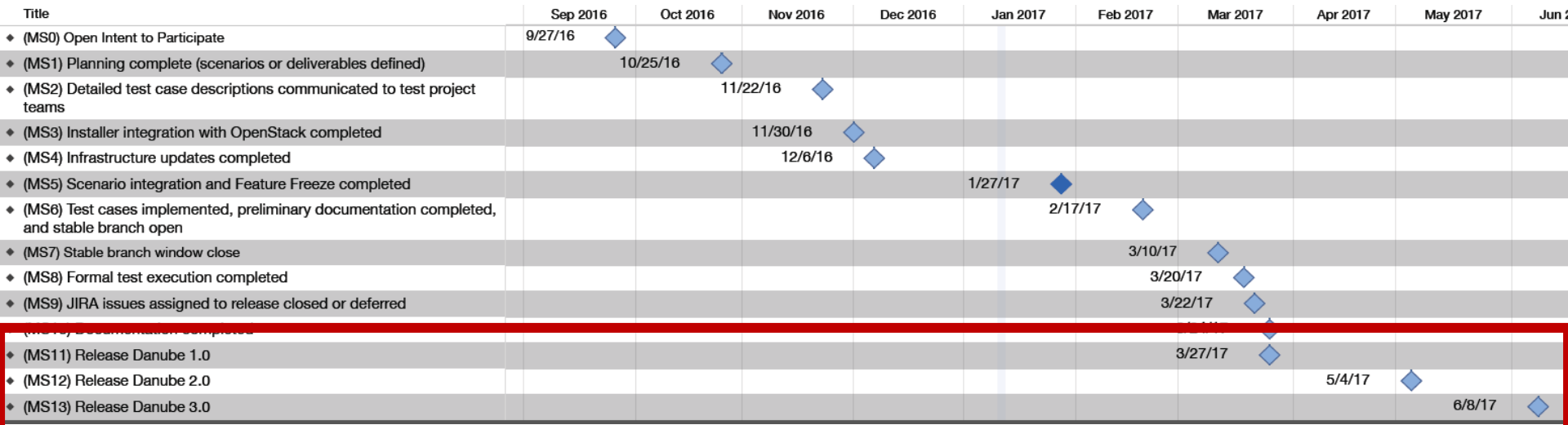
OPNFV Danube Review

- Projects Intending to Participate in the Danube

Project	PTL	Intent to Participate in Danube	Intent to Participate in Colorado
Movie	@Tianran Zhou	Yes	Yes
Resource Scheduler	Mingjiang Li	No	Yes
Promise	@Peter Lee	Yes	Yes
Prediction	@Hai Liu	No	Yes
DPACC	@Louis Fourie	No	Yes
Genesis	@Frank Brockners	No	Yes
Pinpoint	@Adi Molkho	No	Yes
Policy Test	@Keith Burns	No	Yes
CPerf	@Daniel Farrell	No	Yes
ArmBand	@Bob Monkman	Yes	Yes
Daisy4NFV	@Zhijiang Hu	Yes	No
Parser	@Zhipeng (Howard) Huang	Yes	Yes
Qtip	@Yujun Zhang	Yes	No
Multisite	@Chaoyi Huang	Yes	Yes

OPNFV Danube Review

• Release Plan



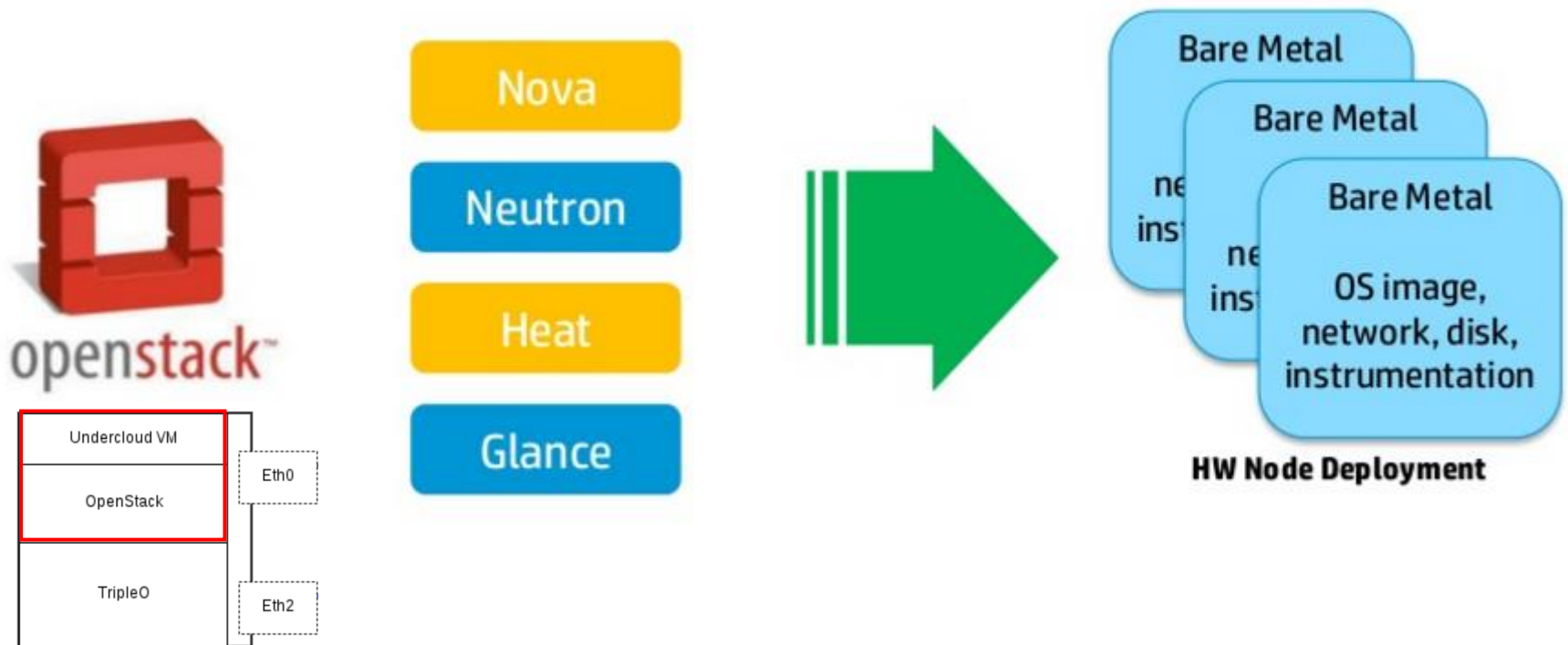
OPNFV Installation-Apex review

OPNFV Installation

- Apex, Compass4Nfv, Fuel or JOID
- Each installer provides the ability to install a **common OPNFV platform** as well as **integrating additional features** delivered through **a variety of scenarios** by the OPNFV community.

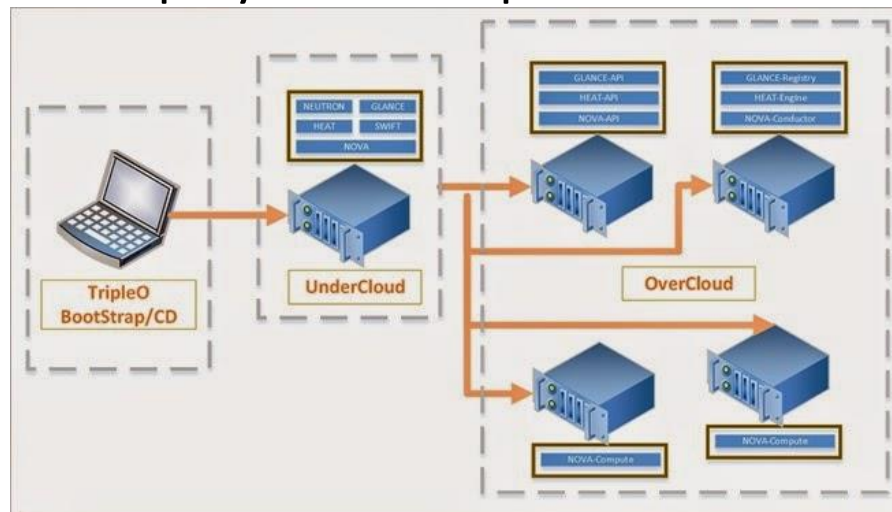
OPNFV Installation-Apex Review

- Triple-O Deployment Architecture
 - Concept of Triple-O: Install OpenStack using OpenStack



OPNFV Installation-Apex Review

- Triple-O Deployment Architecture
 - Triple-O stands for OpenStack on OpenStack
 - OpenStack will be used to install OpenStack
 - Target OPNFV deployment is an OpenStack cloud with NFV features built-in that will be deployed by a smaller all-in-one deployment of OpenStack



OPNFV Installation-Apex Review

- Overview
 - Apex uses the RDO Manager
 - Open Source project as a server provisioning tool
 - RDO Project implementation of OpenStack's Triple-O project
 - RDO : **RPM Distribution of OpenStack(RDO)**
 - Apex deployment toolchains (in “Jumphost”)
 - Apex bootable ISO ([opnfv-apex-danube.iso](#))
 - Install CentOS 7 and the necessary materials to deploy
 - Apex RPM ([opnfv-apex.rpm](#))
 - Installation to a CentOS 7 **libvirt** enabled host
 - RPM contains a collection of configuration file, prebuilt disk images, and the automatic deployment script ([opnfv-deploy](#))

OPNFV Installation-Apex Review

- **Triple-O Deployment Architecture**

- Triple-O stands for OpenStack On OpenStack.
- They are referred to as the undercloud and the overcloud.
- The **undercloud** is used to deploy the overcloud.
 - The undercloud is **the all-in-one installation of OpenStack** that includes baremetal provisioning capability.
- The **overcloud** is OPNFV
 - provision the target OPNFV nodes.

OPNFV Installation-Apex Review

- Apex will deploy three control nodes in an HA deployment. Each of these nodes will run the following services:
 - Stateless OpenStack services
 - MariaDB / Galera
 - RabbitMQ
 - OpenDaylight
 - HA Proxy
 - Pacemaker & VIPs
 - Ceph Monitors and OSDs

OPNFV Installation-Apex Review

- **OPNFV Scenario Architecture**

- The standard naming convention for a scenario is:
< VIM platform>-<SDN type>-<feature>-<ha/noha>

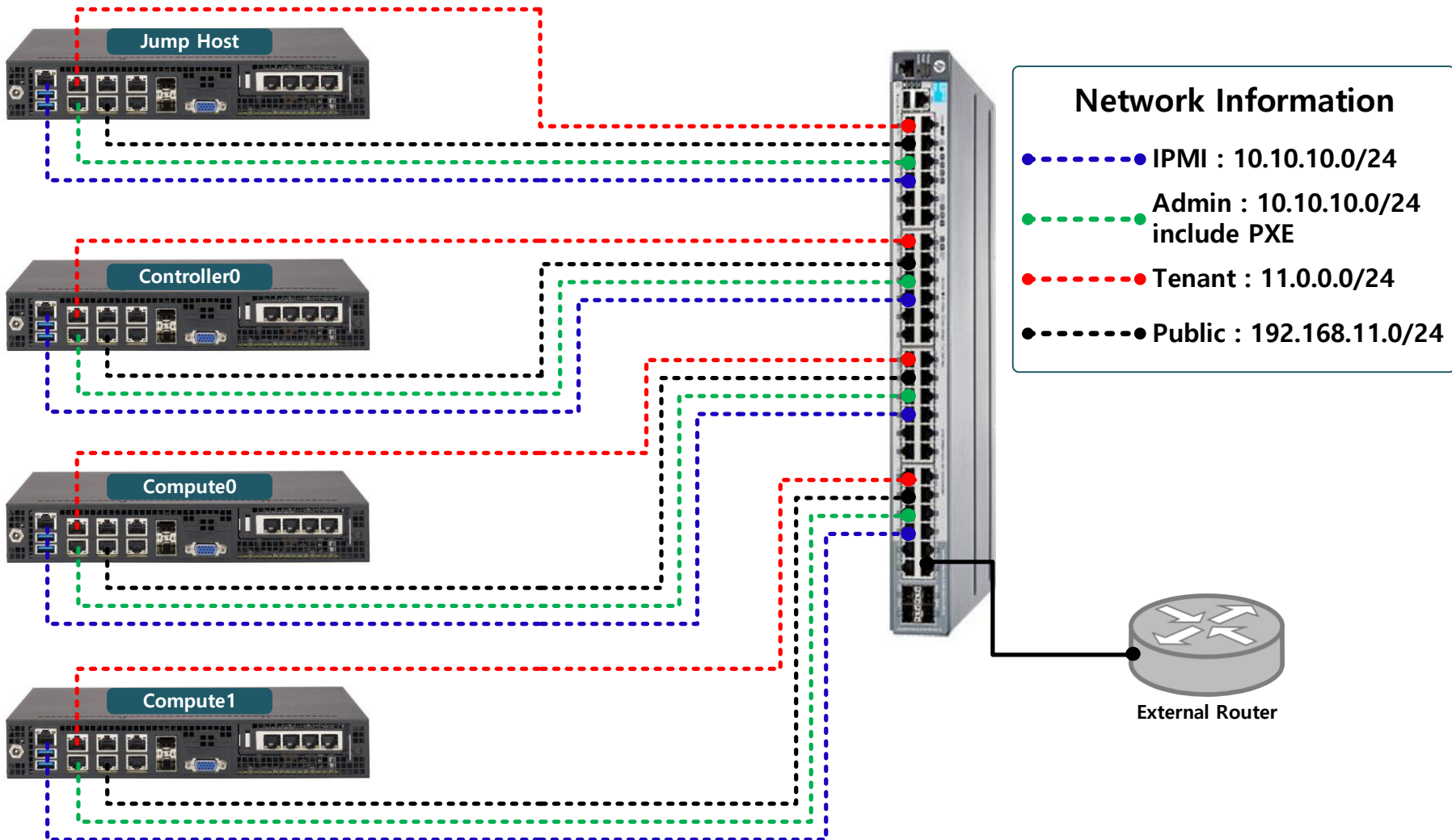
Scenario	Owner	Supported
os-nosdn-nofeature-ha	Apex	Yes
os-nosdn-nofeature-noha	Apex	Yes
os-nosdn-ovs-ha	OVS for NFV	Yes

OPNFV Installation Demo

OPNFV Installation Demo - Environment

- **OPNFV Installation (no HA)**
 - Physical Server Information
 - Intel® Xeon® processor D-1518(8 Core)
 - 64 GB Ram
 - 500G SSD
 - **4 Baremetal**
 - 1 Jump Host
 - 1 Controller Node
 - 2 Compute Node

OPNFV Installation Demo - Environment

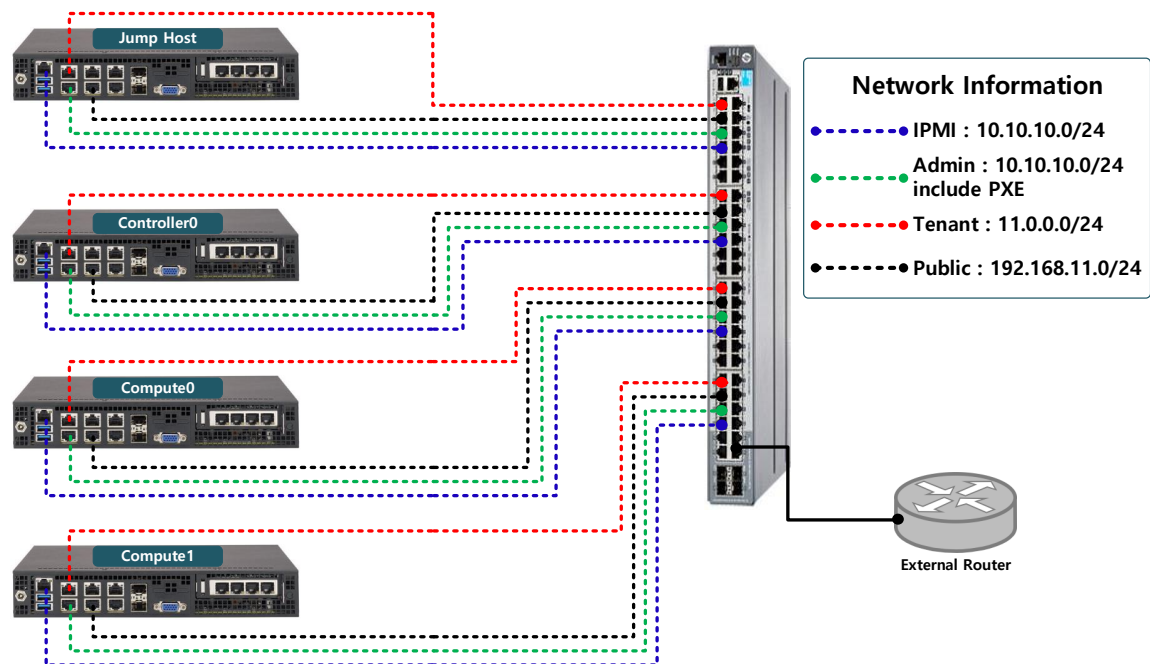


OPNFV Installation Demo - Step

- Install CentOS 7 (minimal version)
- IP Setting (root mode)
- Install Package (Bare Metal Jump host)
- Download apex package and Install
- apex configuration
- apex deploy

OPNFV Installation Demo - Step 1

- IP Setting (root mode)
 - admin configuration
 - admin configuration / tenant network / public network



OPNFV Installation Demo - Step 2

- Install Package (Bare Metal Jumphost)
 - `yum install -y net-tools` : optional → minimal version
 - `yum install -y epel-release`
 - `yum update -y`
 - `yum groupinstall -y "Virtualization Host" chkconfig libvirtd on`
 - → when you reboot, it automatically reboot vm
 - `yum groupinstall -y "X Window System "` → viewer virt manager
 - `yum install -y dejavu-lgc-sans-fonts` → character
 - `yum install -y virt-manager`

OPNFV Installation Demo - Step 3

- Download apex package and Install

OPNFV Mitaka library

- <https://repos.fedorapeople.org/repos/openstack/openstack-mitaka/rdo-release-mitaka-6.noarch.rpm>

OPNFV Apex Package

- <http://artifacts.opnfv.org/apex/colorado>
- `opnfv-apex-3.0-colorado-3.0.noarch.rpm`
- `opnfv-apex-common-colorado-3.0.noarch.rpm`
- `opnfv-apex-undercloud-3.0-colorado-3.0.noarch.rpm`

OPNFV Installation Demo - Step 3

- Download apex package and Install

Dependencies library

- <http://artifacts.opnfv.org/apex/dependencies>
 - python34-markupsafe-0.23-9.el7.centos.x86_64.rpm
 - python3-ipmi-0.3.0-1.noarch.rpm
 - python3-jinja2-2.8-5.el7.centos.noarch.rpm

OPNFV Installation Demo - Step 3

- **Download apex package and Install**

Dependencies library

- <http://artifacts.opnfv.org/apex/dependencies>
 - python34-markupsafe-0.23-9.el7.centos.x86_64.rpm
 - python3-ipmi-0.3.0-1.noarch.rpm
 - python3-jinja2-2.8-5.el7.centos.noarch.rpm

OPNFV Installation Demo - Step 4

- `yum install -y rdo-release-mitaka-6.noarch.rpm`
- `yum install -y python*`
- `yum install -y opnfv-apex-*`

OPNFV Installation Demo - Step 4

* config file

- /etc/opnfv-apex

* install script and image

- /var/opt/opnfv/lib
- /var/opt/opnfv/images

* execute file

- /usr/bin/
 - opnfv-deploy ==> undercloud and overcloud shell
 - opnfv-clean ==> apex undercloud uninstall
 - opnfv-util ==> connect shell

OPNFV Installation Demo - Step 5

- **Installation High-level Overview**
 - **opnfv-deploy** uses three configuration files in order to know how to install and provision the OPNFV target system
 - `/etc/opnfv-apex/inventory.yaml`
 - bare-metal info. (IPMI IP, MAC for PXE, ..)
 - `/etc/opnfv-apex/deploy_settings.yaml`
 - Deployment options
 - `/etc/opnfv-apex/network_settings.yaml`
 - Network requirements
 - **opnfv-deploy** will boot the **undercloud VM** and load the target deployment configuration into the provisioning toolchain.

OPNFV Installation Demo - Step 5

- /etc/opnfv-apex/inventory.yaml

```
nodes:
  node1:
    mac_address: "00:1E:67:E3:1D:E2"
    ipmi_ip: 10.4.17.2
    ipmi_user: root
    ipmi_pass: lovedcn123
    pm_type: "pxe_ipmitool"
    cpus: 2
    memory: 8192
    disk: 40
    arch: "x86_64"
    capabilities: "profile:control"
  node2:
    mac_address: "00:1E:67:E3:1D:E6"
    ipmi_ip: 10.4.17.3
    ipmi_user: root
    ipmi_pass: lovedcn123
    pm_type: "pxe_ipmitool"
    cpus: 2
    memory: 8192
    disk: 40
    arch: "x86_64"
    capabilities: "profile:control"
```

OPNFV Installation Demo - Step 5

- /etc/opnfv-apex/deploy_settings.yaml

```
[root@localhost opnfv]# vi /etc/opnfv-apex/  
inventory.yaml      network_settings.yaml.org  os-nosdn-ovs-ha.yaml      os-odl_12-bgpvpn-ha.yaml  os-odl_12-sfc-noha.yaml  
inventory.yaml.org  os-nosdn-fdio-noha.yaml    os-nosdn-ovs-noha.yaml    os-odl_12-fdio-ha.yaml    os-odl_13-nofeature-ha.yaml  
network_settings_v6.yaml  os-nosdn-nofeature-ha.yaml  os-nosdn-performance-ha.yaml  os-odl_12-fdio-noha.yaml  os-onos-nofeature-ha.yaml  
network_settings.yaml  os-nosdn-nofeature-noha.yaml  os-ocl-nofeature-ha.yaml    os-odl_12-nofeature-ha.yaml  os-onos-sfc-ha.yaml
```

```
global_params:  
  ha_enabled: true  
  
deploy_options:  
  sdn_controller: opendaylight  
  sdn_13: false  
  tacker: false  
  congress: false  
  sfc: true  
  vpn: false
```

OPNFV Installation Demo - Step 5

- /etc/opnfv-apex/network_settings.yaml

```
admin_network:
  enabled: true
  network_type: bridged
  bridged_interface: 'ens513f0'
  bond_interfaces: ''
  vlan: native
  usable_ip_range: 192.0.2.2,192.0.2.99
  #gateway: 192.0.2.1
  #provisioner_ip: 192.0.2.12
  cidr: 192.0.2.0/24
  dhcp_range: 192.0.2.2,192.0.2.10
  introspection_range: 192.0.2.100,192.0.2.120

public_network:
  enabled: true
  network_type: bridged
  bridged_interface: 'enp4s0f0'
  cidr: 192.168.11.0/24
  gateway: 192.168.11.1
  floating_ip_range: 192.168.11.200,192.168.11.220
  usable_ip_range: 192.168.11.10,192.168.11.199
  #provisioner_ip: 192.168.11.12
```

OPNFV Installation Demo - Step 5

- Apex Deploy
- `cd /etc/opnfv-apex/`

```
opnfv-deploy -i inventory.yaml -n  
network_settings.yaml -d os-odl_l2-sfc-noha.yaml
```

DEMO

OPNFV Service Function Chaining(SFC) Tutorial

Contents

- OPNFV SFC Project Introduction
- OPNFV SFC Release History
 - Brahmaputra
 - Colorado
- OPNFV SFC in Danube Release
 - Release Plan
 - Test Cases
- OPNFV SFC Tutorial
 - Overview
 - Installation
 - Demo

OPNFV SFC Project Introduction

- **Project Creation Date:** May 5, 2015
- **Project Category:** Collaborative Development
- **Project Lead:** Brady Johnson
(brady.allen.johnson@ericsson.com)
- **Related Upstream Projects:** OpenStack, ODL, OVS

Committers:

- Brady Johnson (brady.allen.johnson@ericsson.com)
- Manuel Buil (mbuil@suse.com)
- Jim Guichard (jguichar@cisco.com)
- Paul Quinn (paulq@cisco.com)
- Reinaldo Penno (rapenno@gmail.com)
- Vishal Murgai (vmurgai@cisco.com)
- Sam Hague (shague@redhat.com)
- Tim Rozet (trozet@redhat.com)

Contributors:

- Christopher Price (christopher.price@ericsson.com)
- Danny Zhou (danny.zhou@intel.com)
- Johnson Li (johnson.li@intel.com)
- Eric Multanen (eric.w.multanen@intel.com)
- Georgios Paraskevopoulos (geopar@intracom-telecom.com)
- Vijayendra Radhakrishna (vradhakrishna@mvista.com)
- Juan Vidal (juan.vidal.allende@ericsson.com)

OPNFV Wiki: <https://wiki.opnfv.org/display/sfc/Service+Function+Chaining+Home>

Project Mirror in Github: <https://github.com/opnfv/sfc>

OPNFV SFC Release History

- Brahmaputra
 - Requirements in Genesis
 - Preliminary Dependencies
 - OpenDaylight Beryllium release
 - Service Function Chaining
 - Neutron Northbound
 - Group Based Policy
 - Open vSwitch DataBase (OVSDB)
 - OpenFlow Plugin
 - Open vSwitch with NSH support
 - Cisco has branched OVS and created an NSH patch
 - OpenStack Kilo
 - OpenStack Tacker sub-project

OPNFV SFC Release History

- Brahmaputra

The OPNFV SFC Brahmaputra Release ***cannot*** release without these:

- Deploy a complete SFC solution

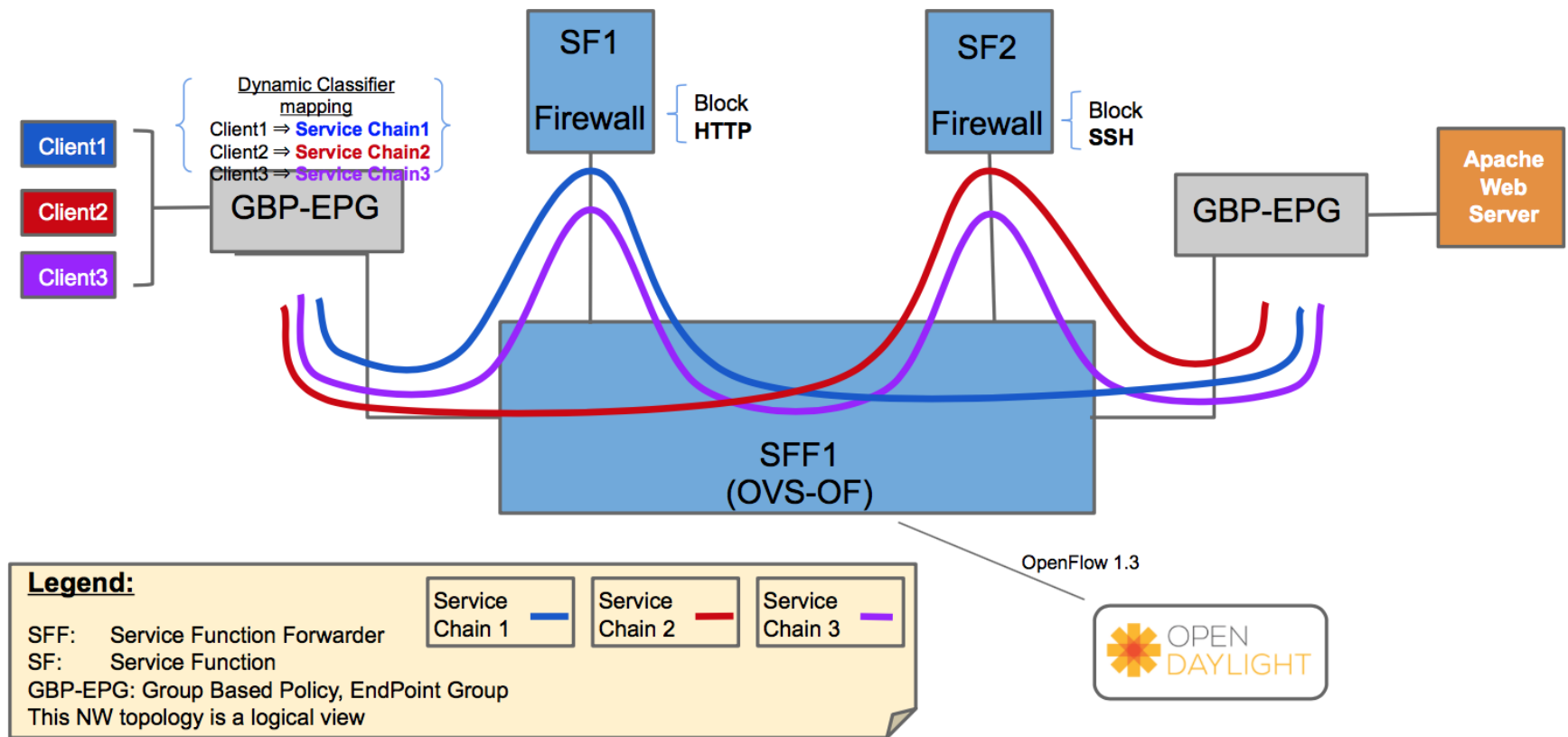
- Create **BGS**/Genesis/Fuel scripts
 - Control node with 2 SFs and the SFF in the host OVS
 - This depends on the SFC ODL/OpenStack coordination (which SFs to create?)
 - Deploy a Compute Node with ODL SFC, OpenStack, and an injector
 - Need to distinguish between initial setup versus rebooting once everything has been setup
 - Needs to allow for multiple Compute Nodes: In the future this will need to be supported

- SFC ODL and OpenStack Coordination

- SFC needs the following info from OpenStack for each Service Function (SF) VM:
 - IP Address
 - encapsulation details (VxLAN, NSH enabled)
 - OVS switch and port the SF is connected to
- SF creation alternatives (the proactive approach will most likely be the way to go)
 - **Reactive approach:** SFC can query OpenStack for SF VM info, but we need to know which VMs to query
 - For example, it may be that 10 VMs are started, but only 2 of them are SFs, which ones do we query?
 - **Proactive approach:** SFC receives the JSON SF config, and for each SF, request OpenStack to create it and return the necessary info. Then SFC will have to complete the SF config with the retrieved VM info.
 - This is only needed for initial SF creation.

OPNFV SFC Release History

- Brahmaputra – Initial use-case



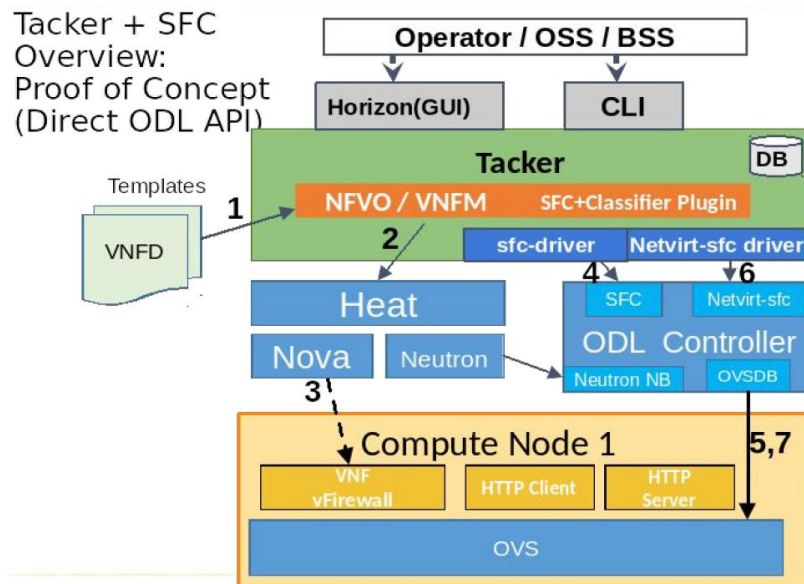
- <https://wiki.opnfv.org/display/sfc/SFC-ODL+Brahmaputra+Testing>

OPNFV SFC Release History

- Colorado
 - OPNFV SFC Scope addition:
 - OPNFV SFC depends on a VNF Manager for a complete end-to-end solution
 - Starting with the Colorado release, we would like for the VNF Manager to be installed via either the Fuel or Apex installation scripts
 - This does not preclude other OPNFV installers
 - The VNF Manager should not be limited to only Tacker, thus allowing for any Open Source VNF Manager to be included
 - In subsequent releases, OPNFV SFC may also use MANO-based Orchestration.

OPNFV SFC Release History

- Colorado
 - Uses OVS 2.5.90 (Intel Patch)
 - OpenDaylight Boron / OpenStack Mitaka
 - OpenStack Tacker project (customized)
 - Direct API communication between Tacker and OpenDaylight




OPNFV SFC Release History

- Colorado – Feature list

1. Better support for multiple compute nodes

- a. Should mainly involve improving Tacker

- b.  **SFC-34** - Improved support for multiple compute nodes **CLOSED**

2. Enhanced Tacker - ODL SFC interactions


- a. This should mainly impact Tacker and possibly ODL Beryllium SR2

- b.  **SFC-35** - Enhanced Tacker ODL SFC interactions **OPEN**

3. Switch to official version of OVS with NSH

- a. Depends on upstream OVS delivery. Hopefully it will be in an OVS 2.5 sub-release.

- b. This will have impacts on Fuel, Apex, and Yardstick

- c.  **SFC-36** - Switch to Yi Yang version of OVS with NSH **CLOSED**

4. Improve the VnfMgrSim

- a. This is contained in the OPNFV SFC code base

- b. This will allow us to use OPNFV SFC (in a reduced manner) without having to depend on Tacker

- c.  **SFC-37** - Improve the VnfMgrSim **OPEN**

<https://wiki.opnfv.org/display/sfc/OPNFV+SFC+Colorado+Release+Plan>



OPNFV SFC Release History

- Colorado – Feature list

- 5. OPNFV SFC testing

- a. Yardstick testing

- b. Functest testing

- c.  ~~SFC-38~~ - Create additional test cases 

- 6. Bare-metal CI executions with the following installers:

- a. Apex

- b. Fuel



- c. Joid (tentative)

- d.  ~~SFC-39~~ - Bare-metal CI executions 

- 7. Improved installation documentation

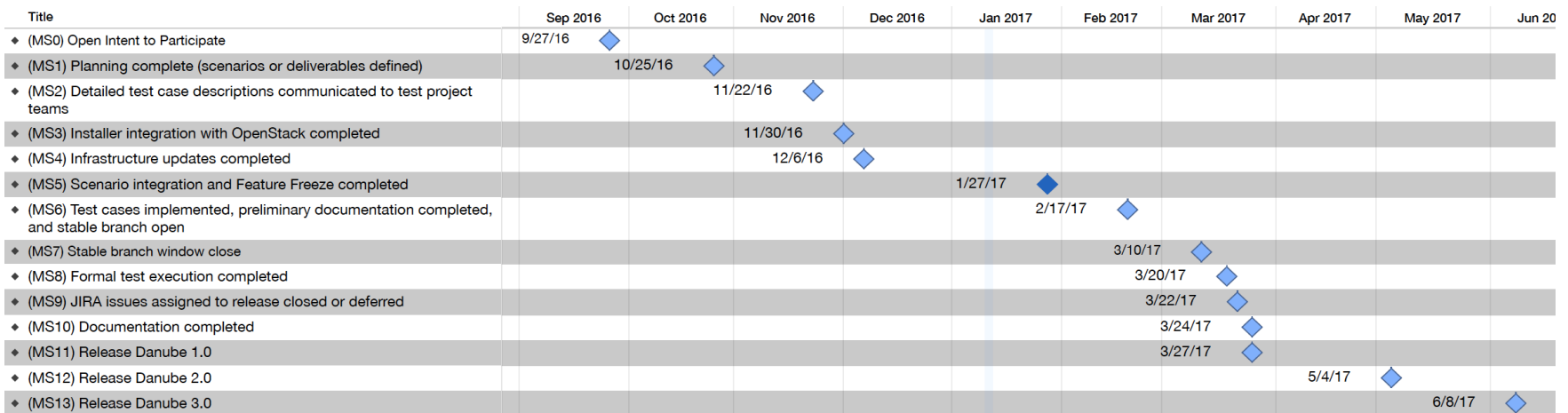
- a.  ~~SFC-40~~ - Improved OPNFV SFC installation documentation 

- 8. OPNFV SFC working with JOID installer (tentative depending on personnel resources)

- a.  ~~SFC-41~~ - OPNFV SFC working with the JOID installer 

OPNFV SFC in Danube Release

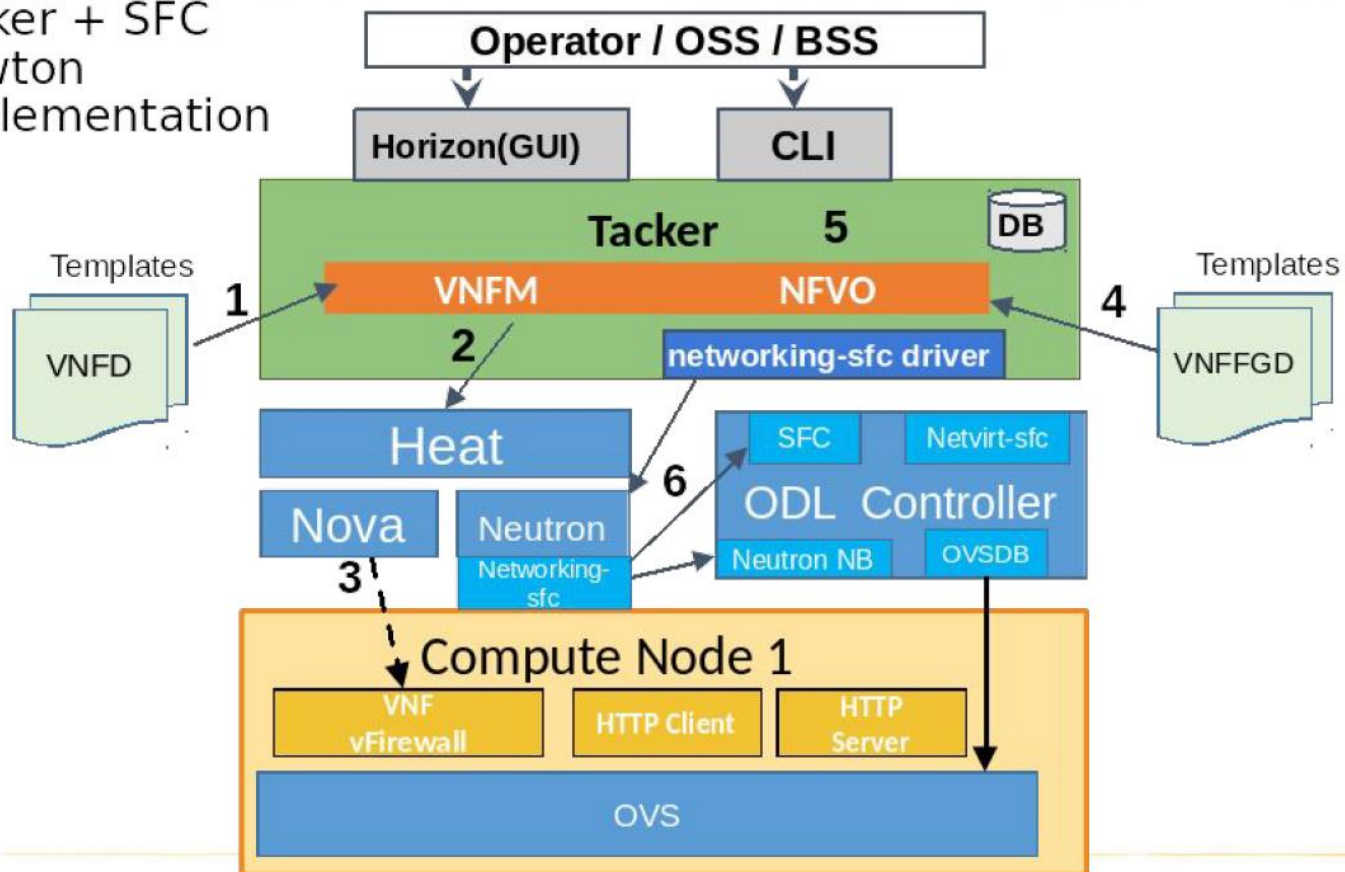
- Release Plan
 - Integration with the OPNFV Fast Data Stacks project to incorporate the FD.io VPP virtual switch
 - Improved High Availability
 - Additional Functest test cases
 - Integrate with Open-O - Time permitting
 - Integrate with OpenStack Gluon - Time permitting



OPNFV SFC in Danube Release

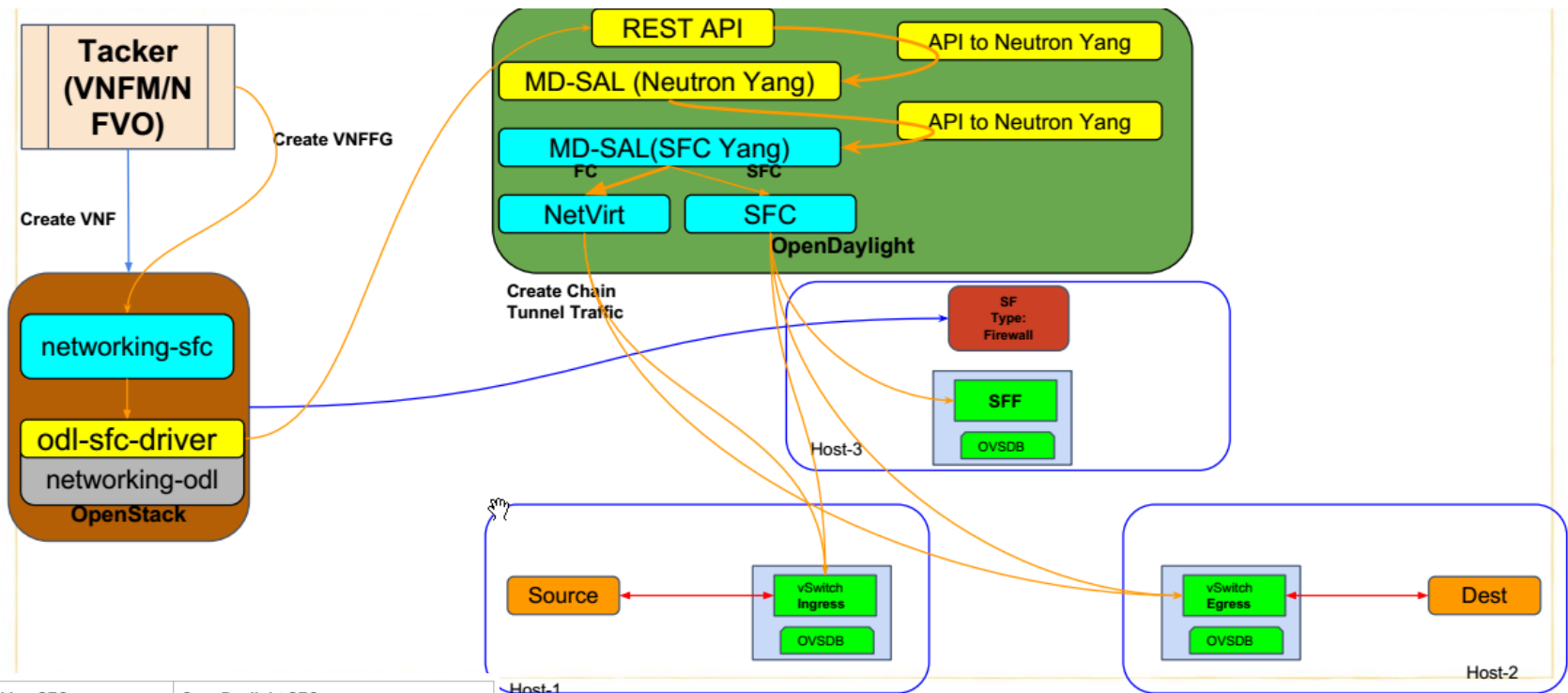
- New OpenStack + ODL Implementation

Tacker + SFC
Newton
Implementation



OPNFV SFC in Danube Release

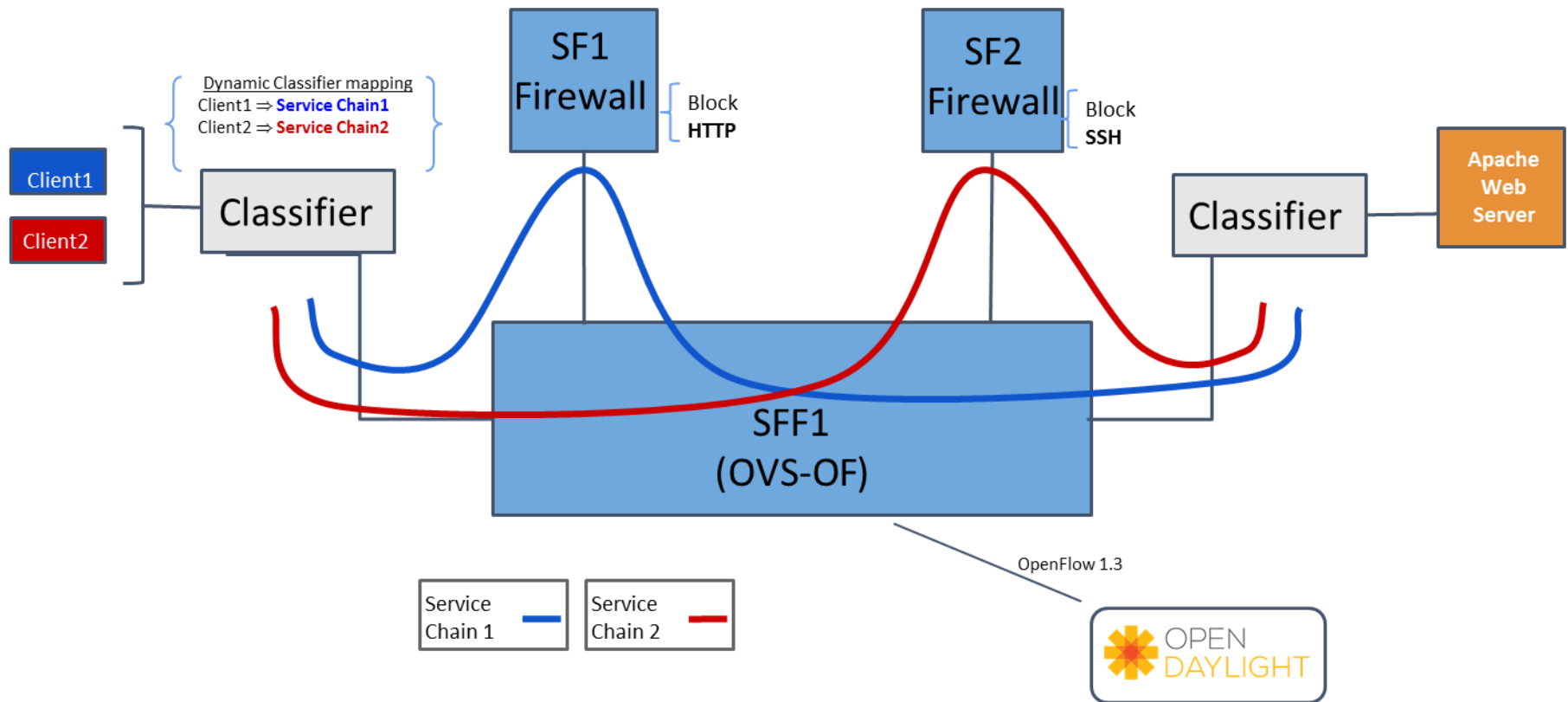
- New OpenStack + ODL Implementation



OpenStack Networking SFC	OpenDaylight SFC
Port Pair	SF/SFF
Port Pair Group	SF/SFF
Port Pair Chain	SFC/SFP/RSP
Flow Classifiers	ACL

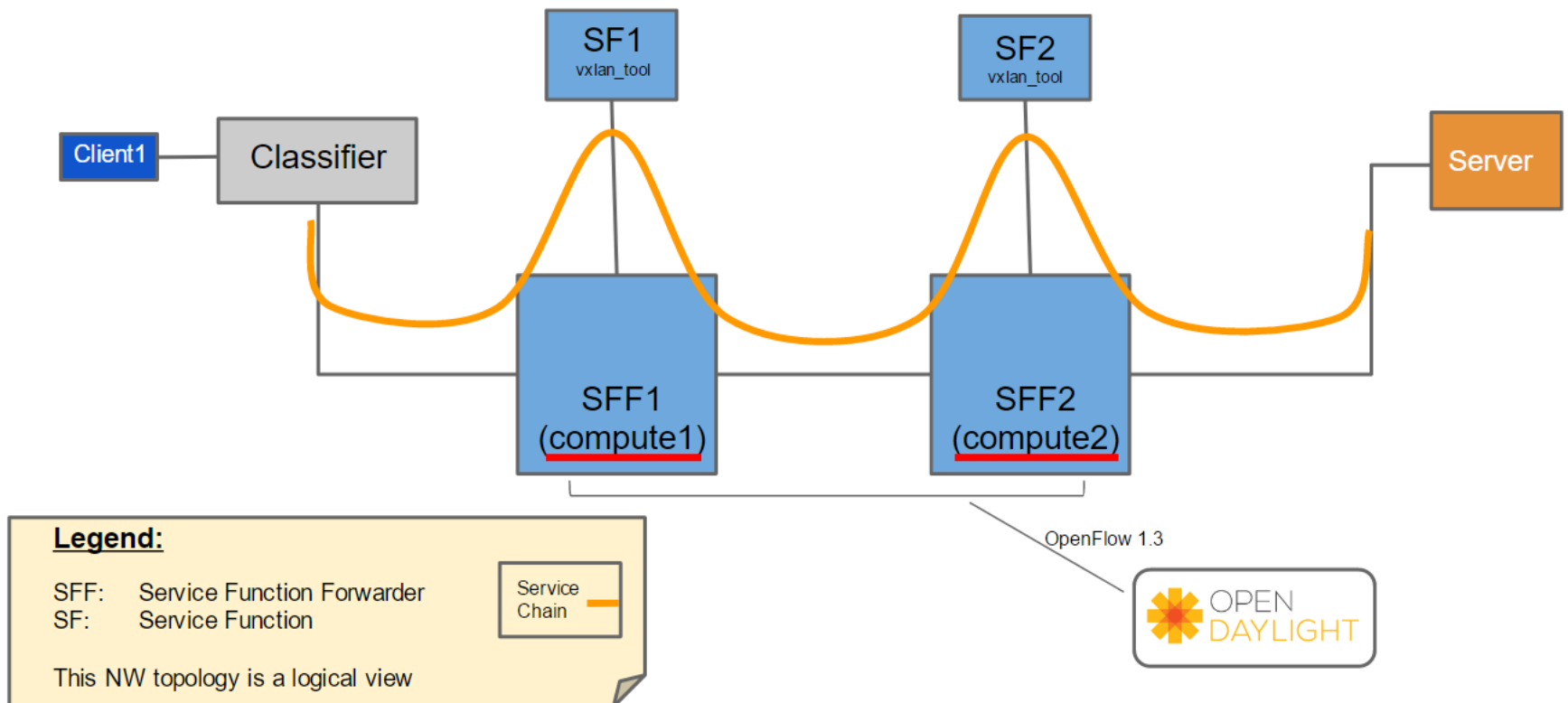
OPNFV SFC in Danube Release

- Checks two chains from service chaining



OPNFV SFC in Danube Release

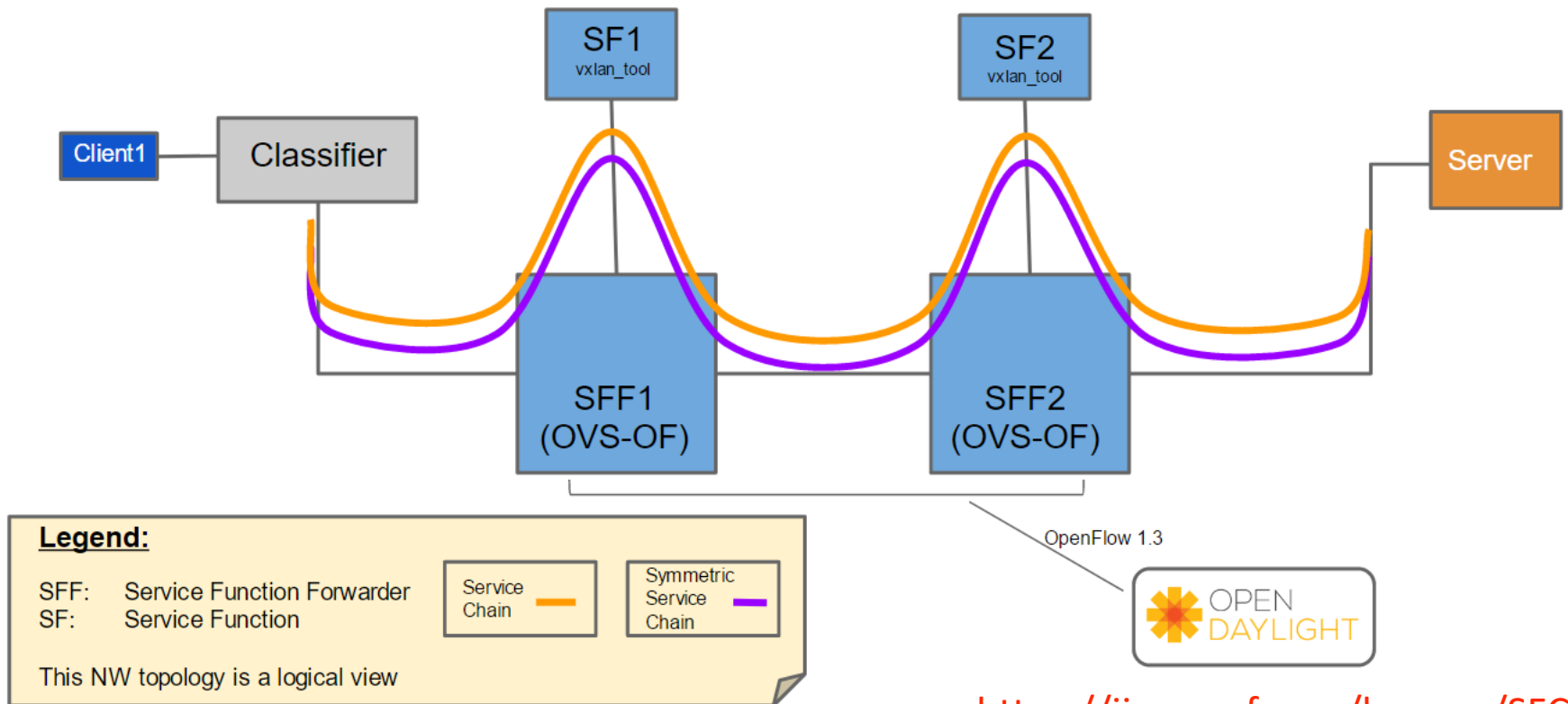
- Test several SF in one chain (SFs in different compute node)



<https://jira.opnfv.org/browse/SFC-55>

OPNFV SFC in Danube Release

- Verify the mechanism of symmetric service chains



<https://jira.opnfv.org/browse/SFC-53>

OPNFV SFC in Danube Release

- SFC Test Challenges

- **Technical:**

- Tacker might still have a bug regarding symmetric chains
 - Symmetric chains can be created but are not tracked as tacker item and thus a classifier cannot be assigned to it
 - How to check that the traffic comes back through the symmetric chain?
 - Client and server in different neutron networks without a router (preferred)
 - Check the counters of openflow tables
 - Check the output of vxlan_tool in SFs
 - Having 2 SFs in the same chain but in different computes
 - Never tested

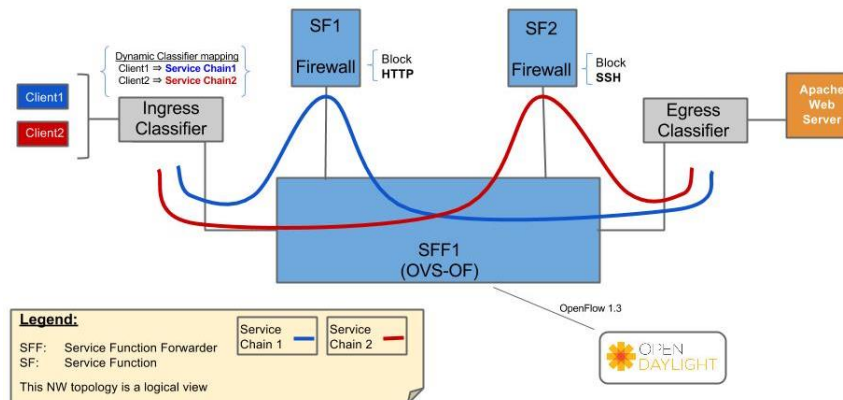
- **Non-technical:**

- Functest changing how projects will call their tests

OPNFV SFC Tutorial

OPNFV SFC Tutorial

- Internal Test Environment
 - OPNFV Installer : Apex (Open-O)
 - 4 Physical Servers : 1-JumpServer, 1-Controller, 2-Compute
 - OPNFV Danube/Colorado version
 - Using Customized OpenStack Tacker (from Redhat)
- Internal Testing
 - Test-case 1 – SFC two chains (SSH and HTTP)



OPNFV SFC Tutorial

- Demo Video

Thank you!